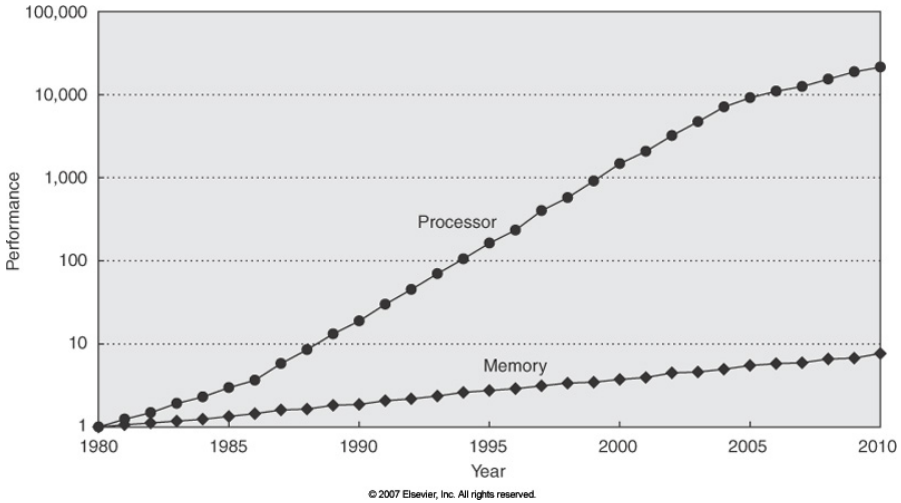
HY425 Lecture 12: Cache Memories

Dimitrios S. Nikolopoulos

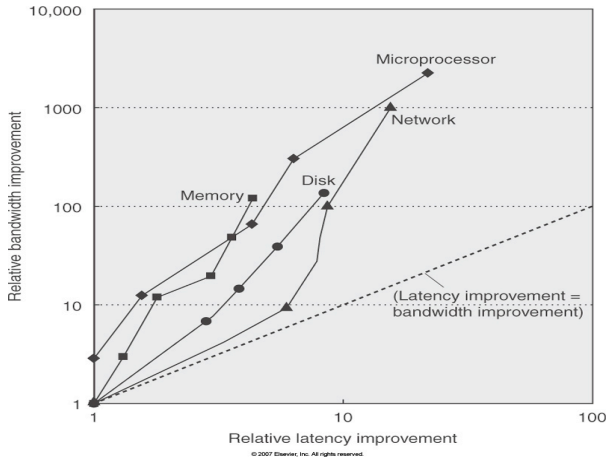
University of Crete and FORTH-ICS

November 23, 2011

Moore's law and processor-memory gap



Latency lags bandwidth

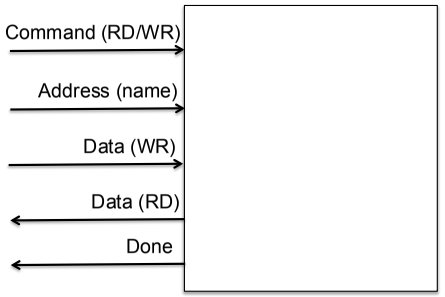


Memory abstraction in architecture

Addressable memory

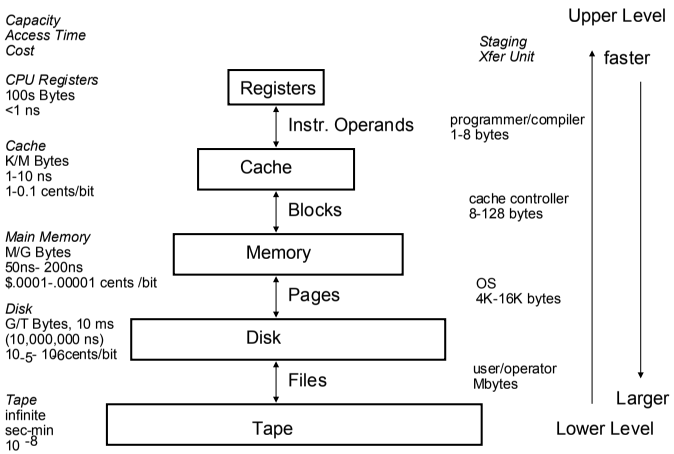
- Association between address and values in storage
- Addresses index bytes in storage
- Values aligned in multiples of word size
- Accessed through sequence of reads and writes
- Write binds value to address
- Read returns most recent value stored in address

Generic memory



Memory hierarchy

Speed vs. cost trade-off



Cache

Definition

- First level of memory hierarchy after registers
- Any form of storage that bufferes temporarily data
  - OS buffer cache, name cache, Web cache, ...
- Designed based on the principle of locality
  - Temporal locality: Accessed item will be accessed again in the near future
  - Spatial locality: Consecutive memory accesses follow a sequential pattern, references separated by unit stride

Locality recap

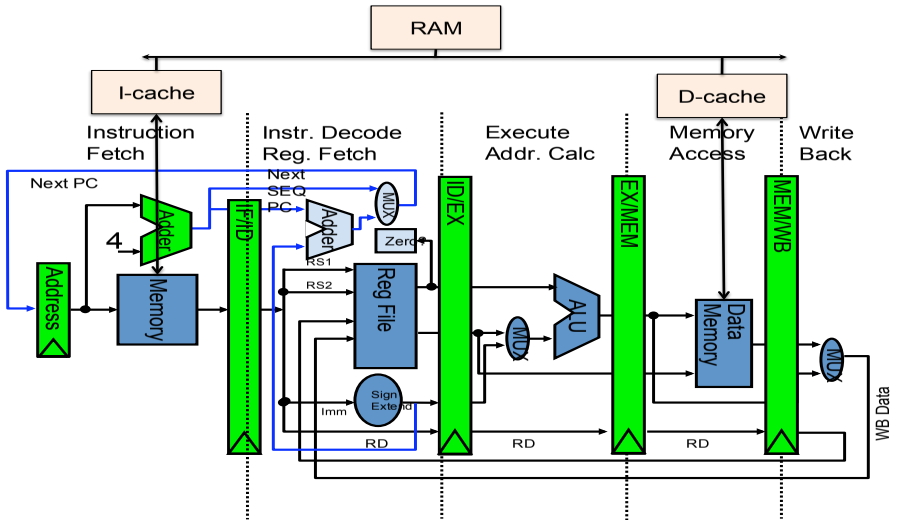
Spatial locality

- Appears due to iterative execution and linear data access patterns
- Exploited by using larger block sizes – data to be used prefetched with block
- Exploited by data and code transformations by the compiler
- Exploited by unit-stride prefetching mechanisms and policies

Temporal locality

- Appears due to iterative execution and data reuse
- Exploited by caches, through which data is reused
- Working set: data that needs to be kept cached in a window of time to maximize locality
- Reuse distance: number of blocks of memory accessed between two consecutive accesses to same block

Caches in 5-stage pipeline



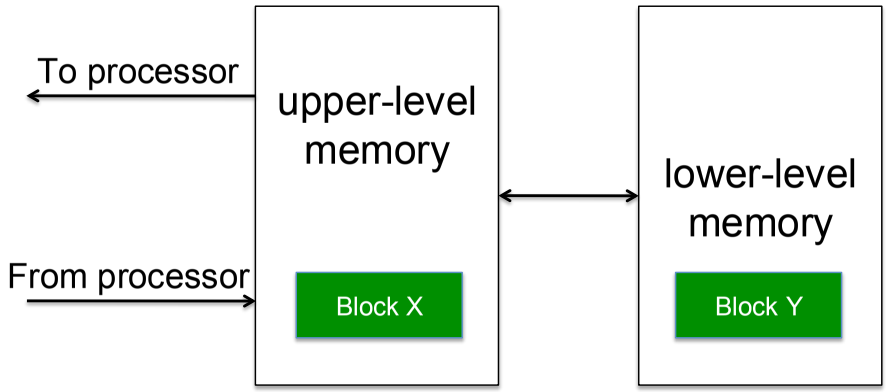
Terminology

Hits and misses

- Hit: data appears in a block in the level of the memory hierarchy searched (e.g. L1 cache)
  - Hit rate: Ratio of accesses to given level of the memory hierarchy that hits
  - Hit time: TIme to deliver block to processor from given level of the memory hierarchy, including time to determine hit or miss and time to access memory
- Miss: data is not found in any block in the level of the memory hierarchy search (e.g. L1 cache) and needs to be searched at the lower level of the memory hierarchy (e.g. L2 cache)
  - Miss rate: 1 - hit rate
  - Miss penalty: Time to load block in the upper level from lower level, plus time to deliver block to the processor

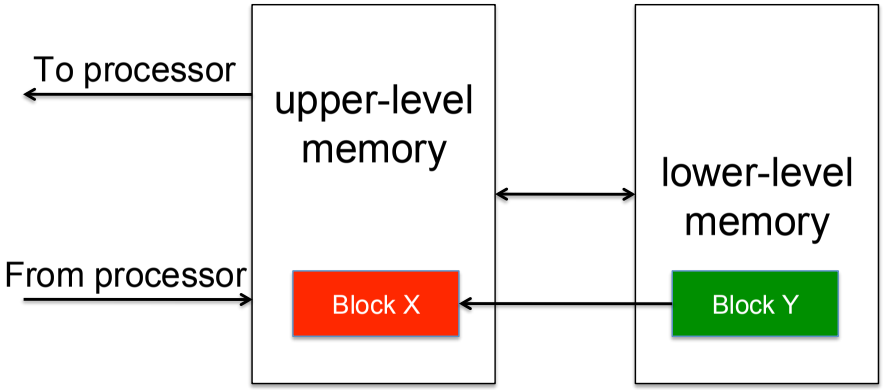
Terminology (cont.)

Cache hit Block X



Terminology (cont.)

Cache miss Block X



3 C's model

Characterization of cache misses

- Compulsory miss: Miss that happens due to the first access to a block since program began execution. Also called cold-start miss.
- Capacity miss: Miss that happens because a block that has been fetched in the cache needed to be replaced due to limited capacity (all blocks valid in the cache, cache needed to select victim block). Block had been fetched, replaced, and re-fetched to count as capacity miss.
- Conflict miss: MIss that happens because address of block maps to same location in the cache with other block(s) in memory. Block had been fetched, replaced, re-fetched, and cache has invalid locations that could hold the block if a different address mapping scheme were used, to count as conflict miss (as opposed to compulsory miss with first-time fetch).

4 Questions for Memory Hierarchy

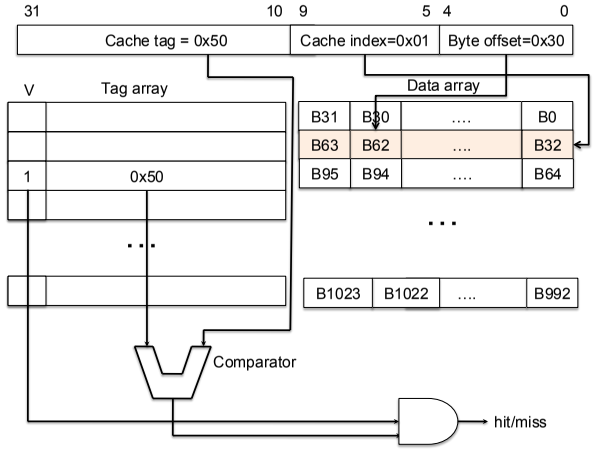
For a given level of the memory hierarchy

- Q1: Where can a block be placed in the upper level? (Block placement)
- Q2: How is a block found if it is in the upper level? (Block identification)
- Q3: Which block should be replaced on a miss? (Block replacement)
- Q4: What happens on a write? (Write strategy)

Direct-mapped cache

Modulo address mapping

1K direct-mapped cache, 32-byte blocks



Direct-mapped cache

Advantages

- Simple, low complexity, low power consumption
- Fast hit time
- Data available before cache determines hit or miss
  - Hit/miss check done in parallel with data retrieval

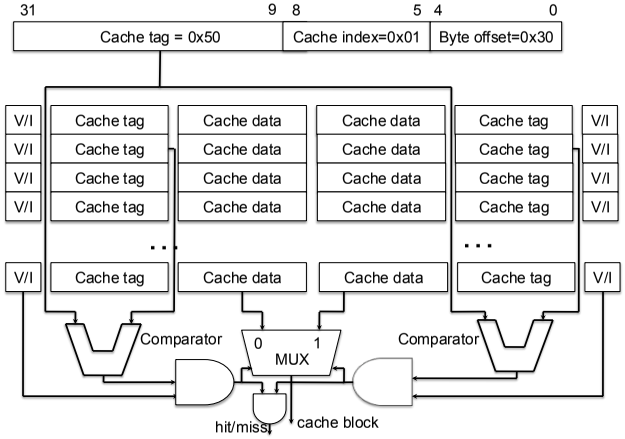
Disadvantages

- Conflicts between blocks mapped to same block in cache

Two-way set associative cache

Modulo address mapping

1K two-way associative cache, 32-byte blocks



Two-way set associative cache

Advantages

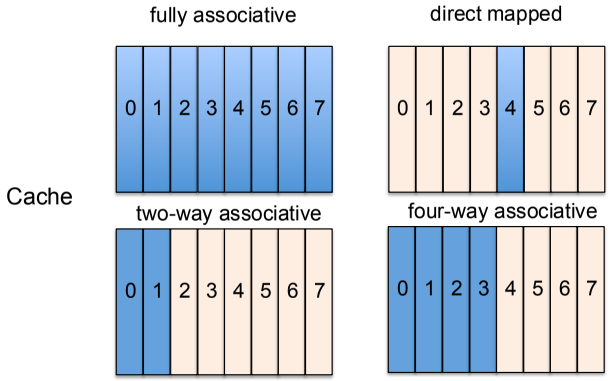
- Choice of mapping memory block to different cache blocks in a set
  - LRU or other policies for good selection of victim blocks
- Reduction of conflicts

Disadvantages

- Increased complexity – comparators, multiplexor, parallel tag comparison
- Increased power consumption
- Increased hit time, due to comparators and multiplexor
- Data available after cache determines hit or miss

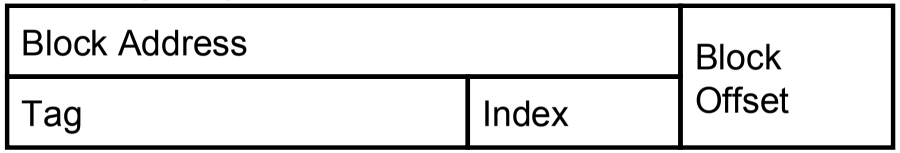
Cache mapping example

Mapping block 12 from RAM in 8-block cache



Q2: how is a block found in the cache

Cache tag array



- Index points to line in data array – one block or set
- Offset points to byte in block
- Tag compared against tag field in address
- Valid bit ORed with output of tag comparator

Q3: which block is replaced on a miss

Common replacement policies

- Random
- Least recently used
  - 2-bit implementation for 2-way associative caches
  - Expensive to implement for high associativity
- FIFO

D-cache misses per 1000 instructions, SPECcpu 2000

	Associativity								
	Two-way			Four-way			Eight-way		
Size	LRU	Random	FIFO	LRU	Random	FIFO	LRU	Random	FIFO
16 KB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

Q4: what happens on a write

Write-through

- Data written to both block in the cache and block in lower-level memory
- Simple to implement, since cache always contains clean data
- Simplified coherence, as lower level always has latest copy of data
- Read misses do not result in writes to lower-level
- Repeated writes to same location incur latency of lower-level memory each
  - Write buffers used to mask latency of lower-level memory

Q4: what happens on a write

Write-back

- Data written only to block in cache
- Modified cache block is written to main memory only when replaced
  - Dirty bit marks block as written since brought in (1) or clean (0)
- Read misses result in writes, if evicted block dirty
- No lower-level latency for repeated writes to same location
- Lower bandwidth consumption, attractive solution for multiprocessors

Q4: what happens on a write

Write miss handling

- Write allocate
  - Block is allocated in cache upon write miss and refilled with new value
  - Write miss behaves like read miss
  - Effective if data is reused by processor for reading
- Write no-allocate
  - No block is allocated in cache, write goes directly to lower-level
- Effective if data is not reused by processor (e.g. write-once streaming data)

Average memory access time (AMAT)

AMAT components

$$\text{Average memory access time} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

$$\text{CPU time} = (\text{CPU execution clock cycles} + \text{Memory stall clock cycles}) \times \text{Clock cycle time}$$

$$\text{CPU time} = IC \times \left(CPI_{execution} + \frac{\text{Memory stall clock cycles}}{\text{Instruction}} \right) \times \text{Clock cycle time}$$

$$\text{CPU time} = IC \times \left(CPI_{execution} + \text{Miss rate} \times \frac{\text{Memory accesses}}{\text{Instruction}} \times \text{Miss penalty} \right) \times \text{Clock cycle time}$$

Example

UltraSPARC III

- in-order processor
- $CPI_{execution}$ = 1.0
- miss penalty = 100 cycles
- miss rate = 2%
- 1.5 memory references per instruction
- 30 cache misses per 1000 instructions

$$\text{CPU time} = IC \times \left(1.0 + \frac{100 \times 30}{1000}\right) \times \text{Clock cycle time} = IC \times 4 \times \text{cycle time}$$

$$\text{CPU time} = IC \times \left(1.0 + 0.02 \times \frac{1.5}{1} \times 100\right) \times \text{Clock cycle time} = IC \times 4 \times \text{cycle time}$$

Example

UltraSPARC III

- Cache miss latency increases execution time by 4*x*
- Higher clock rates imply more clock cycles wasted due to miss penalty
  - Higher relative impact of cache on performance
- HW/SW cache-conscious optimizations attempt reduce AMAT
- Performance depends on both clock cycle and AMAT – trade-off

Example

Direct-mapped vs. set-associative cache

- $CPI_{execution}$ = 2.0
- 64 KB caches with 64-byte blocks
- 1.5 memory references per instruction
- Direct mapped cache miss rate = 1.4%
- Set associative cache stretches clock cycle by 1.25, miss rate = 1.0%
- 1 GHz processor
- 75 ns miss penalty
- 1 cycle hit time

$$AMAT_{direct-mapped} = 1.0 + (.014 \times 75) = 2.05ns$$

$$AMAT_{2-way} = 1.0 \times 1.25 + (.01 \times 75) = 2.00ns$$

Example

Direct-mapped vs. set-associative cache

$$\text{CPU time} = IC \times \left(CPI_{execution} + \frac{Misses}{Instruction} \times \text{miss penalty}\right) \times \text{clock cycle time}$$

$$\text{CPU time}_{direct-mapped} = IC \times (2.0 \times 1.0 + 0.014 \times 1.5 \times 75) = 3.58 \times IC$$

$$\text{CPU time}_{two-way} = IC \times (2.0 \times 1.25 + 0.01 \times 1.5 \times 75) = 3.63 \times IC$$

- Associative cache achieves lower AMAT than direct-mapped cache
- Direct-mapped cache achieves higher performance than associative cache

Overlapping memory latency in OOO processors

Miss penalty in OOO

- Processor can execute instructions while cache miss is pending
- Processors can execute instructions also while cache hit is pending
- Hard to attribute stall cycles to instructions
  - Stall cycle is any cycle where at least one instruction does not commit
  - First

$$\frac{\text{Memory stall cycles}}{\text{instruction}} = \frac{\text{Misses}}{\text{instruction}} \times (\text{Total miss latency} - \text{overlapped miss latency})$$

Improving cache performance

4 strategies

- Reducing miss penalty: multilevel caches, critical word first, read miss before write miss, merging write buffers, victim caches
- Reducing miss rate: larger block size, larger cache size, higher associativity, way prediction and pseudo associativity, compiler optimizations
- Reducing miss rate or miss penalty via parallelism: non-blocking caches, hardware prefetching, compiler prefetching
- Reducing hit time: small and simple caches, avoiding address translation, pipelined cache access, trace caches